

Fast L1-NMF for Multiple Parametric Model Estimation*

Mariano Tepper
Guillermo Sapiro

*Department of Electrical and Computer Engineering
Duke University*

MARIANO.TEPPER@DUKE.EDU
GUILLERMO.SAPIRO@DUKE.EDU

Abstract

In this work we introduce a comprehensive algorithmic pipeline for multiple parametric model estimation. The proposed approach analyzes the information produced by a random sampling algorithm (e.g., RANSAC) from a machine learning/optimization perspective, using a *parameterless* biclustering algorithm based on L1 nonnegative matrix factorization (L1-NMF). The proposed framework exploits consistent patterns that naturally arise during the RANSAC execution, while explicitly avoiding spurious inconsistencies. Contrarily to the main trends in the literature, the proposed technique does not impose non-intersecting parametric models. A new accelerated algorithm to compute L1-NMFs allows to handle medium-sized problems faster while also extending the usability of the algorithm to much larger datasets. This accelerated algorithm has applications in any other context where an L1-NMF is needed, beyond the biclustering approach to parameter estimation here addressed. We accompany the algorithmic presentation with theoretical foundations and numerous and diverse examples.

Keywords: Multiple parametric model estimation, robust fitting, RANSAC, nonnegative matrix factorization, biclustering, L1 optimization

1. Introduction

This paper addresses the problem of fitting multiple instances of a given parametric model to data corrupted by noise and outliers. This is a prevalent problem for example in computer vision, found in a wide range of applications such as finding lines/circles/ellipses in images, homography estimation in stereo vision, motion estimation and segmentation, and the geometric analysis of 3D point clouds. Finding a single instance of a parametric model in a dataset is a robust fitting problem that is hard on its own; the difficulties are exacerbated when multiple instances might be present in the dataset due to the unavoidable emergence of pseudo-outliers (data points that belong to one structure and are usually outliers to any other structure). Thus, we face the problem of simultaneous robust estimation of model parameters and attribution of data points to the estimated models. These two problems are intrinsically intertwined. Furthermore, the correct number of model instances is not known in advance.

Formally, the data is a set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^m$ of m objects, described by

$$\mathcal{X} = \left(\bigcup_k \mathcal{X}_k \right) \cup \mathcal{O} \quad \text{with} \quad (\forall k) \mathcal{X}_k \cap \mathcal{O} = \emptyset. \quad (1)$$

*. This work was partially supported by Google, NSF, ONR, NGA, and ARO.

The objects in each subset \mathcal{X}_k , which might intersect, are (noisy) measurements that can be described with a parametric model $\mu(\theta_k)$, with parameter vector θ_k . In the following, we say that the objects in \mathcal{X}_k are inliers to the model $\mu(\theta_k)$. We also generally refer to a set of objects as inliers, in the sense that there is a statistically meaningful instance of a parametric model that describes them. The objects in \mathcal{O} cannot be described with such a model and we refer to them as outliers.

Let us define more formally these intuitive concepts. A model μ is defined as the zero level set of a smooth parametric function $f_\mu(\mathbf{x}; \theta)$,

$$\mu(\theta) = \{\mathbf{x} \in \mathbb{R}^d, f_\mu(\mathbf{x}; \theta) = 0\}, \quad (2)$$

where θ is a parameter vector. We define the error associated with the datum $\mathbf{x} \in \mathcal{X}$ with respect to the model $\mu(\theta)$ as

$$e_\mu(\mathbf{x}, \theta) = \min_{\mathbf{x}' \in \mu(\theta)} \text{dist}(\mathbf{x}, \mathbf{x}'), \quad (3)$$

where dist is an appropriate distance function. Using this error metric, we define the Consensus Set (CS) of a model as

$$\mathcal{C}_\mu(\mathcal{X}, \theta, \delta) = \{\mathbf{x} \in \mathcal{X}, e_\mu(\mathbf{x}, \theta) \leq \delta\}, \quad (4)$$

where δ is a threshold that accounts for the measurement noise and/or model mismatch.

Multiple Parametric Model Estimation (MPME) seeks to find the set of inliers-model pairs $(\mathcal{X}_k, \theta_k)$ from the observed \mathcal{X} such that $\mathcal{X}_k = \mathcal{C}_\mu(\mathcal{X}, \theta_k, \delta)$. This problem is, by itself, ill-posed. It is standard in the literature to implicitly or explicitly impose a penalty on the number of recovered pairs to render it tractable. We also adopt such a design choice. Notice that once \mathcal{X}_k is found, the corresponding θ_k can be estimated for example by simple least-squares regression, i.e.,

$$\hat{\theta}_k = \underset{\theta}{\operatorname{argmin}} \sum_{\mathbf{x} \in \mathcal{X}_k} [e_\mu(\mathbf{x}, \theta)]^2. \quad (5)$$

MPME is an important but difficult problem, as standard robust estimators, like RANSAC (RANDOM SAMPLE CONSENSUS) [8, 13], are designed to extract a single model. Let us then begin by formally explaining how does the RANSAC machinery work, to further illustrate the value and perspective of our proposed MPME formulation.

Let us denote by b the minimum number of elements necessary to uniquely characterize a given parametric model, e.g., $b = 2$ for lines and $b = 3$ for circles. For example, if we want to discover alignments in a 2D point cloud, the elements are 2D points, models μ are lines, and $b = 2$, since a line is defined by two points. A set of b objects is called a minimal sample set (MSS). Generically, the random-sample-and-model-generation framework can be described by Algorithm 1. We first randomly sample n MSSs, each generating one model hypothesis. The number n is an overestimation of the number of trials needed to obtain a certain number of “good” models [13, 31, 37]. Then, we compute the CS of each model hypothesis using Equation (4). Let \mathcal{U} be the set of all these consensus sets. From this point onwards, different algorithms perform different operations on \mathcal{U} . Specifically, RANSAC, the

Algorithm 1: Random sampling algorithm

input : set of objects \mathcal{X} , parametric function f_μ , inliers threshold δ .
output: pool \mathcal{U} of consensus sets.

- 1 $b \leftarrow$ minimum number of elements necessary to uniquely characterize model μ , see Equation (2);
- 2 **foreach** $j \in \{1 \dots n\}$ **do**
- 3 Select at random a minimal sample set (MSS) $\mathcal{X}_{\text{MMS}(j)}$ of size b from \mathcal{X} ;
- 4 Estimate θ_j from $\mathcal{X}_{\text{MMS}(j)}$ by solving $(\forall \mathbf{x} \in \mathcal{X}_{\text{MMS}(j)}) f_\mu(\mathbf{x}; \theta) = 0$;
- 5 $\mathcal{U} \leftarrow \{\mathcal{C}_\mu(\mathcal{X}, \theta_j, \delta)\}_{j=1}^n$, see Equation (4);

algorithm that introduced this framework, detects a single model by taking the CS from \mathcal{U} with the largest size, and uses it to estimate θ from \mathcal{C} as in Equation (5).

Applying RANSAC sequentially, removing the inliers from the dataset as each model instance is detected, has been proposed as a solution for multi-model estimation, e.g., [24]. However, this approach is known to be suboptimal [37]. The multiRANSAC algorithm [37] provides a more effective alternative, although the number of models must be known a priori, imposing a very limiting constraint in many applications. An alternative approach consists of finding density modes in a parameter space. The overall idea is that we can map the data into the parameter space through random sampling, and then seek the modes of the distribution by discretizing the distribution, i.e., using the Randomized Hough Transform [35], or by using non-parametric density estimation techniques, like the mean-shift clustering algorithm [27]. These, however, are not intrinsically robust techniques, even if they can be robustified with outliers rejection heuristics [31]. Moreover, the choice of the parametrization and its discretization are critical, among other important shortcomings [31]. The computational cost of these techniques can be very high as well. From a different perspective, J-linkage [31], T-linkage [20], and RPA [21] address the problem by clustering the dataset. We will provide more details about these methods in the next section.

Contributions. We have previously introduced a novel framework and perspective to reach consensus in grouping problems by re-framing them as biclustering problems [30]. In particular, the task of finding/fitting multiple parametric models to a dataset was, for the first time, formally posed as a consensus/biclustering problem. In this work, we build upon this framework, introducing a complete and comprehensive algorithmic pipeline for multiple parametric model estimation. The proposed approach preserves and considers all the information produced by Algorithm 1 (i.e., no averaging/pooling is performed). This information is then analyzed from a machine learning/optimization perspective, using a *parameterless* biclustering algorithm based on nonnegative matrix factorization (NMF).

Contrarily to the main trends in the literature, the proposed modeling does not impose non-intersecting subsets \mathcal{X}_i . Secondly, it exploits consistencies that naturally arise during the RANSAC execution, while explicitly avoiding spurious inconsistencies. This new formulation conceptually changes the way that the data produced by the popular RANSAC, and related model-candidate generation techniques, is analyzed.

With respect to our previous work [30, Section 5], the present work includes the following key differences and improvements:

- We have streamlined the pre- and post-processing steps of the biclustering algorithm, simplifying algorithmic choices and unifying the pipeline for different types of parametric models;
- No user intervention (i.e., no parameter tuning) is required to find the number of parametric models. During the biclustering process, this number is automatically determined using model selection techniques (i.e., minimum description length);
- We accelerate the biclustering process introducing an accelerated algorithm to solve L1-based nonnegative matrix factorization (L1-NMF) problems. This allows to solve medium-sized problems faster while also extending the usability of the algorithm to much larger datasets. This contribution exceeds the context of this work, as this algorithm has potential applications in any other context where an L1-NMF is needed (e.g., traffic analysis [29], eldercare [29], shadow removal for face detection [36], video surveillance [36]).

We provide the complete source code of the proposed approach for generating all the examples presented in this work.¹

Organization. The remainder of this paper is organized as follows. In Section 2 we present the proposed *biclustering* approach for multiple parametric model estimation. In Section 3 we describe the fast version of the biclustering algorithm. We present diverse and extensive experimental results in Section 4 and finally provide some closing remarks in Section 5.

2. Random Sample Ensemble

Let \mathbf{X} be a matrix. In the following, $(\mathbf{X})_{ij}$, $(\mathbf{X})_{:j}$, $(\mathbf{X})_{i:}$ denote the (i, j) -th entry of \mathbf{X} , the j -th column of \mathbf{X} , and the i -th row of \mathbf{X} , respectively.

The input of the algorithm is a pool $\mathcal{U} = \{\mathcal{C}_\mu(\mathcal{X}, \theta_j, \delta)\}_{j=1}^n$ of consensus sets (the output of Algorithm 1).

Definition 1 (Preference matrix) *We define the $m \times n$ preference matrix \mathbf{A} , whose rows and columns represent respectively the $m = |\mathcal{X}|$ data elements $\{\mathbf{x}_i\}_{i=1}^m$ and the $n = |\mathcal{U}|$ consensus sets, as*

$$(\mathbf{A})_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}_i \in \mathcal{C}_\mu(\mathcal{X}, \theta_j, \delta); \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

An example of a preference matrix for a synthetic dataset can be seen in Figure 1. Traditionally, in algorithms like RANSAC, the preference matrix is (often implicitly) analyzed column-by-column or row-by-row. The preference matrix was explicitly introduced in the context of MPME in [31]. In the original formulation, the objects in \mathcal{X} were clustered using the rows of \mathbf{A} as feature vectors, obtaining a powerful state-of-the-art technique called J-linkage. An extension to work with a non-binary (using soft versus hard element-model membership) version of \mathbf{A} was proposed in [20].

1. <https://github.com/marianotepper/arise>

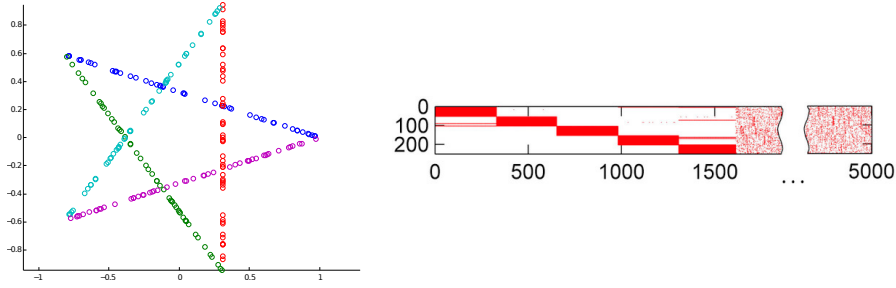


Figure 1: Preference matrix example. The data (objects) consist of 250 points on five line segments (models) forming a star. The rows of the preference matrix \mathbf{A} were reordered (permuted) by group for improved visualization.

An alternative approach involves creating an $m \times m$ co-occurrence matrix, defined as

$$\mathbf{B} = \frac{1}{n} \sum_{k=1}^n \mathbf{B}_k, \quad \text{where} \quad (\mathbf{B}_k)_{ij} = \begin{cases} 1 & \text{if } x_i, x_j \in \mathcal{C}_\mu(\mathcal{X}, \theta_k, \delta); \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

The matrix \mathbf{B} is widely used in the context of ensemble clustering where it is built from clusters obtained from multiple clustering algorithms, instead of consensus sets. There are, within this research area, many algorithms to analyze \mathbf{B} : from simple techniques such as applying a clustering algorithm to it (e.g., k -means, hierarchical or spectral clustering), to more complex matrix factorization techniques [e.g., 19]. It is important to point out that \mathbf{A} contains more information than \mathbf{B} . Noticing that

$$\mathbf{B} = \frac{1}{n} \sum_{j=1}^n (\mathbf{A})_{:j} (\mathbf{A})_{:j}^T, \quad (8)$$

it becomes clear that the averaging operation implies the loss of critical information.

In the vast majority of matrix factorization/clustering methods, the number of factors/clusters is a critical and hard-to-tune parameter. But this is in fact one of the parameters we are interested in discovering! In the context of MPME, matrix factorization has been recently applied [21] to a normalized version of \mathbf{B} (using soft membership). Provided that the correct number of factors is selected, this method delivers state-of-the-art results.

An additional constraint common to all the aforementioned clustering/factorization approaches is that they provide a segmentation of the dataset elements. This means that these models do not correctly handle intersecting/overlapping models. As we will see next, this limitation is not present in the proposed formulation.

2.1 Analyzing the preference matrix

Our approach follows a different conceptual path, simultaneously analyzing the rows and columns of the preference matrix \mathbf{A} . To explain the rationale behind our formulation, let us rewrite Equation (2) for clarity purposes:

$$\mathcal{X} = \left(\bigcup_{k=1}^r \hat{\mathcal{X}}_k \right) \cup \mathcal{O} \quad \text{with} \quad (\forall k) \quad \hat{\mathcal{X}}_k \cap \mathcal{O} = \emptyset.$$

Algorithm 2: Random Sample Ensemble

input : set of objects \mathcal{X} , parametric function f_μ , inliers threshold δ .
output : collection of inliers-model pairs $\{(\mathcal{C}_t, \theta_t)\}_{t=1}^T$.

- 1 Execute Algorithm 1 to obtain \mathcal{U} ;
- 2 Form the preference matrix \mathbf{A} from \mathcal{U} (Definition 1);
- 3 Obtain T biclusters $\{(\mathbf{u}_t, \mathbf{v}_t)\}_{t=1}^T$ from \mathbf{A} (see Section 2.1 for details, note that T is automatically estimated);
- 4 **foreach** \mathbf{u}_t **do**
- 5 $\mathcal{C}_t \leftarrow \{x_i \in \mathcal{X} \mid (\mathbf{u}_t)_i > 0\}$;
- 6 Estimate θ_t from \mathcal{C}_t using least-squares, see Equation (5);
- 7 Re-compute \mathcal{C}_t as $\mathcal{C}_\mu(\mathcal{X}, \theta_t, \delta)$, see Equation (4);

Each $\hat{\mathcal{X}}_k$ represents one of the *ground truth* groups that we are seeking to recover. We assume $(\forall k) |\hat{\mathcal{X}}_k| \geq b$, where b is the minimum number of elements necessary to uniquely characterize a given parametric model.

Let us assume that during the execution of Algorithm 1 we obtained several pure MSSs for some $\hat{\mathcal{X}}_k$, i.e., $\mathcal{J}_k = \{j \mid \mathcal{X}_{\text{MMS}(j)} \subseteq \hat{\mathcal{X}}_k\}$ (the index j corresponds to the iterations of Algorithm 1). Then, for *almost all* $j, j' \in \mathcal{J}_k$ their respective models will be very similar, i.e., $\theta_j \approx \theta_{j'}$; these models should also be similar to the model estimated from $\hat{\mathcal{X}}_k$ using least squares. Therefore their consensus sets should be similar, i.e., $\mathcal{C}_\mu(\mathcal{X}, \theta_j, \delta) \approx \mathcal{C}_\mu(\mathcal{X}, \theta_{j'}, \delta) \approx \hat{\mathcal{X}}_k$. From the definition of \mathbf{A} , we can then write that

$$(\mathbf{A})_{:\mathcal{J}_k} = \hat{\mathbf{u}}_k \mathbf{1}^\top + \mathbf{O}_k, \quad (9)$$

where $\hat{\mathbf{u}}_k$ is a binary vector such that $(\forall j \in \mathcal{J}_k) (\mathbf{A})_{:j} \approx \hat{\mathbf{u}}_k$, $\mathbf{1}$ is the “all ones” vector of size $|\mathcal{J}_k|$, and \mathbf{O}_k accounts for all the errors in this approximation. Finally, as we look to represent all columns of \mathbf{A} , we can rewrite the above equation, adding the corresponding zeros, as $\mathbf{A} = \hat{\mathbf{u}}_k \hat{\mathbf{v}}_k^\top + \mathbf{O}_k$, where $\hat{\mathbf{v}}_k$ is a binary vector of size n such that $(\forall j \in \mathcal{J}_k) (\hat{\mathbf{v}}_j)_i = 1$ and \mathbf{O}_k is now of size $m \times n$.

We can then extend this representation to all ground truth groups $\hat{\mathcal{X}}_k$, by writing

$$\mathbf{A} = \sum_{k=1}^r \hat{\mathbf{u}}_k \hat{\mathbf{v}}_k^\top + \mathbf{O}, \quad (10)$$

where $\hat{\mathbf{u}}_k \in \{0, 1\}^m$, $\hat{\mathbf{v}}_k \in \{0, 1\}^n$ and again \mathbf{O} accounts for all errors in the approximation. Notice that Equation (10) allows to cast the problem of analyzing the output of Algorithm 1 in terms of a biclustering problem.

Once the biclusters of \mathbf{A} have been identified, we can obtain the consensus sets and, from them, estimate the final output models. See Algorithm 2 for a full description of this process. The next question now is: how do we find each bicluster?

2.2 L1-NMF

As usual in the optimization literature, the problem of finding $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ in Equation (10) is easier to solve if we soften the binary constraints, imposing nonnegativity instead. The only missing component to formalize the problem is an appropriate prior for \mathbf{O} . Since the

errors are also binary (and thus spurious), a reasonable choice would be to penalize its L1 norm. For convenience, we also drop the binary constraint on \mathbf{O} . We can thus write the single-bicluster estimation problem as

$$\min_{\mathbf{u}, \mathbf{v}} \|\mathbf{A} - \mathbf{u}\mathbf{v}^T - \mathbf{O}\|_F^2 \quad \text{s.t.} \quad \mathbf{u} \geq \mathbf{0}, \mathbf{v} \geq \mathbf{0}, \|\mathbf{O}\|_1 \leq \sigma, \quad (11)$$

which, for some σ , is equivalent to our proposed formulation

$$\min_{\mathbf{u}, \mathbf{v}} \|\mathbf{A} - \mathbf{u}\mathbf{v}^T\|_1 \quad \text{s.t.} \quad \mathbf{u} \geq \mathbf{0}, \mathbf{v} \geq \mathbf{0}. \quad (\text{L1-NMF})$$

We use the L1 norm to cope with the impulsive nature of the errors in \mathbf{A} . This formulation computes a robust median approximation to the preference matrix \mathbf{A} , which carries all the needed information. Any standard NMF algorithm can be adapted to use the L1 norm and solve (L1-NMF). The algorithm in [30, App. A] delivers high-quality solutions at the expense of a higher computational cost. In this work, for the sake of speed, we use the following procedure:

1. Find initializations for \mathbf{u} and \mathbf{v} using the iterative re-weighting scheme in [15]. In our experiments, this method proved to be rather fast for small-scale problems, but more inaccurate than other alternatives. This makes it a good choice for initialization.
2. Given the initialization \mathbf{u} , solve the convex problem $\min_{\mathbf{v} \geq \mathbf{0}} \|\mathbf{D}_{\mathbf{u}} (\mathbf{A} - \mathbf{u}\mathbf{v}^T)\|_1$, using the ADMM technique in [30, App. A] ($\mathbf{D}_{\mathbf{x}}$ is a diagonal matrix with the indicator vector $\mathbf{1}_{[\mathbf{x} > 0]}$ in its diagonal).
3. Given the new value of \mathbf{v} , solve the convex problem $\min_{\mathbf{u} \geq \mathbf{0}} \|(\mathbf{A} - \mathbf{u}\mathbf{v}^T) \mathbf{D}_{\mathbf{v}}\|_1$, using the same technique as before.

The method in [15] is not particularly well suited for large-scale problems, as it deals with *dense* weighting matrices with the size of \mathbf{A} . As this size increases, handling these matrices becomes computationally prohibitive. Section 3 is devoted to present a new technique to accelerate the above algorithm, rendering it capable of handling large-scale instances and producing a novel efficient L1-NMF solver.

2.3 Dealing with multiple biclusters

A challenge with biclustering (as with classical clustering) is that the number of biclusters is not an easy parameter to set or to estimate in advance. Following a standard approach in the literature [22, 32] we sieve the information in a sequential way. Generically, we iterate two steps until some stopping criterion is met: (1) find one bicluster (\mathbf{u}, \mathbf{v}) , and (2) subtract the information encoded by (\mathbf{u}, \mathbf{v}) from \mathbf{A} .

Algorithm 3 summarizes the proposed biclustering approach. In Line 6 we set to zero the columns corresponding to the active set of \mathbf{v}_t . This enforces disjoint active sets between the successive \mathbf{v}_t , and hence orthogonality. This also ensures that non-negativity is maintained throughout the iterations. The proposed algorithm is very efficient, simple to code, and demonstrated to work well in the experimental results that we will present later.

The iterations should stop (1) when \mathbf{A} is empty (Line 3), or (2) when \mathbf{A} only contains noise (no structured patterns). We deal with the second case in Line 7, considering that any bicluster formed by a single model instance is spurious.

Algorithm 3: Sequential rank-one biclustering algorithm.

input : matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$
output : set of biclusters $\{(\mathbf{u}_t, \mathbf{v}_t)\}_{t=1}^T$
1 $\mathbf{A}_1 \leftarrow \mathbf{A}$;
2 **foreach** $t \in \{1 \dots n\}$ **do**
3 **if** $\mathbf{A}_t = \mathbf{0}$ **then**
4 \perp break;
5 Find one bicluster $(\mathbf{u}_t, \mathbf{v}_t)$ of \mathbf{A}_t , where $\mathbf{u}_t \in \mathbb{R}_+^m$ and $\mathbf{v}_t \in \mathbb{R}_+^n$;
6 $(\forall i, j) (\mathbf{A}_{t+1})_{ij} \leftarrow \begin{cases} 0 & \text{if } (\mathbf{v}_t)_j > 0, \\ (\mathbf{A}_t)_{ij} & \text{otherwise;} \end{cases}$
7 **if** $\|\mathbf{v}_t\|_0 \leq 1$ **then**
8 \perp break;
9 **if** $t < n$ **then** // if early stop, $t-1$ is the last good bicluster
10 $T \leftarrow t-1$;
11 Prune $\{(\mathbf{u}_t, \mathbf{v}_t)\}_{t=1}^T$ using a model selection strategy;

2.4 Minimum description length as a stopping criterion

The core of Algorithm 3 (lines 1 to 8) does not provide a *reliable* mean to determine the number of biclusters to be extracted from the preference matrix. Indeed, lines 3 and 7 only provide a *rough* stopping criterion when \mathbf{A}_{t+1} is empty or when the bicluster is not the product of a consensus between at least two models. This criterion will output more models than needed, and this section is devoted to prune the collection $\{(\mathbf{u}_t, \mathbf{v}_t)\}_{t=1}^T$.

For each bicluster, we are only interested in the support of its vectors \mathbf{u}_t and \mathbf{v}_t (which are sparse by design). We can then binarize them to avoid being misled by small numerical inaccuracies that might have occurred during the optimization procedure, i.e., $\underline{\mathbf{u}}_t = \lfloor \mathbf{u}_t \rfloor_\gamma$ and $\underline{\mathbf{v}}_t = \lfloor \mathbf{v}_t \rfloor_\gamma$, where

$$(\lfloor \mathbf{x} \rfloor_\gamma)_i = \begin{cases} 1 & \text{if } (\mathbf{x})_i > \gamma \cdot \|\mathbf{x}\|_\infty; \\ 0 & \text{otherwise.} \end{cases} \quad (12)$$

In practice, we set $\gamma = 10^{-4}$ once and for all the tests in this paper.

We now pose the following model selection problem: given the biclusters $\{(\mathbf{u}_t, \mathbf{v}_t)\}_{t=1}^T$ and the remainders $\{\mathbf{A}_{t+1}\}_{t=1}^T$ (which are binary by definition), find the value K in $[1, T]$ such that the preference matrix \mathbf{A} is *optimally* described by $\mathbf{A} \approx \sum_{t=1}^K \mathbf{u}_t \mathbf{v}_t^T + \mathbf{A}_{K+1}$. This can be considered as an hypothetical compression problem where the task is to encode \mathbf{A} using these elements. The model selection procedure then keeps the value K that minimizes the combined codelengths L (in bits) of its components,

$$\operatorname{argmin}_{1 \leq K < T} \sum_{t=1}^K L(\mathbf{u}_t) + \sum_{t=1}^K L(\mathbf{v}_t) + L(\mathbf{A}_{K+1}). \quad (13)$$

Under the usual assumption that \mathbf{u}_t , \mathbf{v}_t , and \mathbf{A}_{K+1} are individually decorrelated, we can describe each one as a (one dimensional) i.i.d. Bernoulli sequence of values. For a p -

dimensional vector \mathbf{p} , this can be efficiently described using an enumerative code [10],

$$L(\mathbf{p}) = \log_2 \binom{p}{\|\mathbf{p}\|_0} + \log_2 p, \quad (14)$$

where $\|\cdot\|_0$ denotes number of non-zero elements of the argument and $\binom{a}{b}$ is the binomial coefficient. With a slight abuse of notation, for a matrix \mathbf{M} we write $L(\mathbf{M}) = L(\text{vec}(\mathbf{M}))$, where $\text{vec}(\cdot)$ is a vectorization operator.

2.5 Statistical validation for pre-processing

The standard random sampling approach (Algorithm 1) to multiple model estimation generates many good model instances (composed of inliers), but also generates many bad models (composed mostly of outliers). In general, the number of bad models exceeds by far the number of good ones. It is not worth devoting computational effort to the analysis of these columns of \mathbf{A} . Any pattern-discovery technique, such as the biclustering approach presented in the previous section, would benefit from having a simple, efficient, and statistically meaningful method for discarding bad models. These models will typically contain only a handful of objects. The question is how do we determine the minimum size for a good consensus set? This important computational contribution, based on the a contrario testing mechanism presented in depth in [12], is addressed next.

Let us assume that we have a set \mathcal{X} of random elements. Let $\mu(\theta)$ be a model with an associated consensus set $\mathcal{C}_\mu(\mathcal{X}, \theta, \delta)$, built with tolerance δ (Equation (4), Page 2). We will assume, under the background model, that all objects are i.i.d. and that the error in Equation (3) *locally* follows an uniform distribution; this type of simple approximations has proven successful for outlier rejection [12]. Let κ be a locality parameter. If there are $|\mathcal{C}_\mu(\mathcal{X}, \theta, \delta)|$ elements at a distance δ from $\mu(\theta)$, we expect to have in average κ times more elements at a distance $\kappa\delta$. We are interested in computing the probability that $\mathcal{C}_\mu(\mathcal{X}, \theta, \delta)$ has at least m_δ elements given that $\mathcal{C}_\mu(\mathcal{X}, \theta, \kappa\delta)$ contains $m_{\kappa\delta}$ elements. The probability of such an event is $\mathcal{B}(m_{\kappa\delta} - b, m_\delta - b; p)$, where \mathcal{B} is the binomial tail and $p = \delta/\kappa\delta = \kappa^{-1}$ is the probability that a random object belongs to the consensus set $\mathcal{C}_\mu(\mathcal{X}, \theta, \delta)$ given that it belongs to $\mathcal{C}_\mu(\mathcal{X}, \theta, \kappa\delta)$. Recall that b is the minimum number of elements necessary to uniquely characterize a given parametric model. As such, b elements in $\mathcal{C}_\mu(\mathcal{X}, \theta, \kappa\delta)$ are necessarily non-random, since they were used to estimate $\mu(\theta)$. This is the reason behind subtracting b elements from $m_{\kappa\delta}$ and m_δ .

Definition 2 Let $\mu(\theta)$ be a model instance, $\mathcal{C}_\mu(\mathcal{X}, \theta, \delta)$ be its associated consensus set, obtained with precision parameter δ , and $\kappa > 1$ be a locality parameter. We define the number of false Alarms (NFA) of model instance $\mu(\theta)$ as

$$\text{NFA}_{\mu, \delta, \kappa}(\mathcal{X}, \theta) = N_{\text{tests}} \cdot \mathcal{B}(|\mathcal{C}_\mu(\mathcal{X}, \theta, \kappa\delta)| - b, |\mathcal{C}_\mu(\mathcal{X}, \theta, \delta)| - b; \kappa^{-1}), \quad (15)$$

where $N_{\text{tests}} = \binom{m}{b}$ represents the total number of possible models. We say that the model $\mu(\theta)$ is said to be ε -meaningful if

$$\text{NFA}_{\mu, \delta, \kappa}(\mathcal{X}, \theta) < \varepsilon. \quad (16)$$

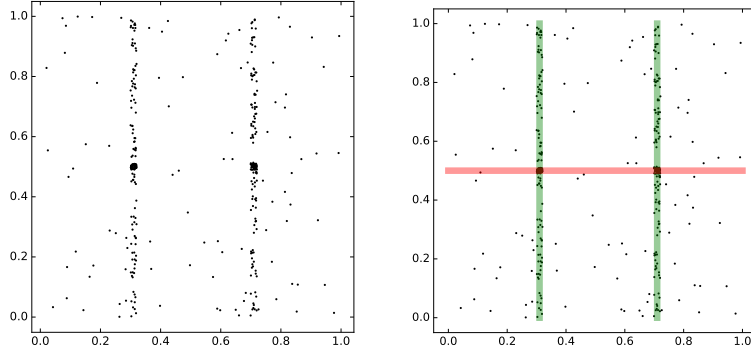


Figure 2: The necessity of an exclusion principle. On the left, an arrangement of points built from uniformly sampling 100 points in $[0, 1]^2$, 100 points in $[0.29, 0.31] \times [0, 1]$, 100 points in $[0.69, 0.71] \times [0, 1]$, 50 in $[0.29, 0.31] \times [0.49, 0.51]$, and 50 points in $[0.69, 0.71] \times [0.49, 0.51]$. On the right, three plausible detections from a hypothetical MPME algorithm. We would like to keep the green lines and discard the red one, which is merely an artifact stemming from the definition of a consensus set (Equation (4)). Note that, after removing the intersection with the green line, there is nothing that distinguishes the red band from the background.

It is easy to prove, by the linearity of expectation, that the expected number of ε -meaningful models in a finite set of random models is smaller than ε . Alternatively, N_{tests} can be empirically set by analyzing a training dataset [6], providing a tighter bound for the expectation.

Definition 2 provides a formal probabilistic method for testing if a model is likely to happen at random or not. From a statistical viewpoint, the method goes back to multiple hypothesis testing. Following an a contrario reasoning [12], we decide whether the event of interest has occurred if it has a very low probability of occurring by chance in the above defined random (background) model. In other words, a model instance $\mu(\theta)$ is ε -meaningful if $|\mathcal{C}_\mu(\mathcal{X}, \theta, \delta)|$ is sufficiently large to have $\text{NFA}_{\mu, \delta, \kappa}(\mathcal{X}, \theta) < \varepsilon$. We only keep columns of \mathbf{A} corresponding ε -meaningful model instances. We set $\varepsilon = 1$ for all the experiments.

As a result of this statistical validation procedure, the preference matrix \mathbf{A} is considerably shrunk (the actual size reduction will depend on the inliers-outliers ratio and the number of model instances in the dataset). This shrunk preference matrix is fed to the biclustering algorithm (Algorithm 3), gaining in stability of the results as well as in speed.

Note 1 *There are techniques in the literature [e.g., 7], which follow a different route: instead of sampling first and prune later, they try to dynamically sample good models from the start. These techniques have proven successful in reducing the number of required samples but operate at a much slower rate per sample. Their dynamic nature makes parallelization more difficult and limited. In our view, a detailed comparison between these paradigms is needed, with special consideration given to the use of parallelization (e.g., GPUs).*

2.6 Statistical validation for post-processing

Once the biclustering algorithm has returned a collection of inliers-model pairs, we need to verify that these models are statistically meaningful from a geometric point of view. For this, we use the test in Definition 2 once again.

Algorithm 4: Exclusion principle

input : collection of inliers-model pairs $\{(\mathcal{C}_t, \theta_t)\}_{t=1}^T$.
output : pruned collection of inliers-model pairs $\{(\mathcal{C}_t, \theta_t)\}_{t=1}^T$.

```
1  $\mathcal{K} \leftarrow \emptyset$ ;  
2  $\mathcal{S} \leftarrow \mathcal{X}$ ;  
3 foreach  $(\mathcal{C}_t, \theta_t)$  do  
4   if  $\text{NFA}_\mu(\mathcal{S}, \theta, \delta)$  is meaningful then  
5      $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\mathcal{C}_t, \theta_t)\}$ ;  
6      $\mathcal{S} \leftarrow \mathcal{S} \setminus \mathcal{C}_t$ ;
```

Additionally, models are allowed to overlap and can thus share elements, which makes situations like the one described in Figure 2 commonplace. In short, a good model should separate itself from the background noise *regardless* of its intersection with other models. The exclusion principle described in [12, Chapter 6] performs this check. It states that, given two groups obtained by the same detector, no point \mathbf{x} is allowed to belong to both groups. In our case, since we are explicitly modeling overlaps between groups, we use a milder version of this principle: we simply ask that any point can only contribute to the NFA of at most one group. This is explained in Algorithm 4. Notice that the order in which the inliers-models pairs are explored affects the result; we sort the biclusters according to their total size $\|\mathbf{u}_t\|_0 \cdot \|\mathbf{v}_t\|_0$ (Equation (12)).

Optionally and if required by the application, as the last step in our post-processing chain, we can force the models to have an empty intersection. There are many alternative ways to address this assignment problem. In this work, we simply assign elements in the intersection of several models to the closest model in distance (see Equation (3)).

3. Accelerated Random Sample Ensemble

The problem of multiple parametric model estimation does not escape the current trend of growth in datasets size, which exposes the need of fast techniques to cope with these massive datasets. Thus, after having described the proposed Random Sample Ensemble algorithmic pipeline in Section 2, we now turn our attention to its acceleration. There are two computational bottlenecks in our approach.

The first one is shared by virtually all the algorithms in the field: running Algorithm 1 (Page 3). Fortunately, all the random samples can be computed in parallel, reducing the problem to clever software engineering. Alternatively, there are recent techniques to reduce the number of needed samples [7], at the expense of a less parallelizable algorithm.

Second, the main component of the proposed technique is the biclustering algorithm, and this section is thus devoted to describe how to efficiently solve this optimization problem. Let us remind the reader that we are seeking to solve Problem (L1-NMF) (Page 7), a challenge with applications beyond the problem at hand. Before moving forward with the exposition, we need to lay the ground by providing a few definitions and key concepts.

The fast Cauchy transform. The fast Johnson-Lindenstrauss transform [1, 2] provides a way to build a low dimensional embedding in the ℓ_2 case and has been widely used in many practical settings. Its ℓ_1 -based analog is the fast Cauchy transform (FCT) [9], which

defines an $h \times m$ embedding matrix $\mathbf{\Pi}$ ($h \ll m$) as

$$\mathbf{\Pi} = 4\mathbf{B}\mathbf{C}\tilde{\mathbf{H}}. \quad (17)$$

The matrices \mathbf{B} , \mathbf{C} , and $\tilde{\mathbf{H}}$ are built such that

- each column of $\mathbf{B} \in \mathbb{R}^{h \times 2m}$ is chosen at random from the h standard basis vectors for \mathbb{R}^h ;
- $\mathbf{C} \in \mathbb{R}^{2m \times 2m}$ is a diagonal matrix with entries sampled from a Cauchy distribution; and
- $\tilde{\mathbf{H}} \in \mathbb{R}^{2m \times m}$ is a block-diagonal matrix comprised of m/s equal blocks along the diagonal (we set $s = h^6$ and we assume it to be a power of two and m/s is an integer), i.e.,

$$\tilde{\mathbf{H}} \stackrel{\text{def}}{=} \begin{array}{|c|c|c|c|} \hline \mathbf{G}_s & & & \\ \hline & \mathbf{G}_s & & \\ \hline & & \ddots & \\ \hline & & & \mathbf{G}_s \\ \hline \end{array} \quad \text{where} \quad \mathbf{G}_s \stackrel{\text{def}}{=} \begin{bmatrix} s^{-1/2} \cdot \mathbf{H}_s \\ \mathbf{I}_s \end{bmatrix}, \quad (18)$$

\mathbf{I}_s is the $s \times s$ identity matrix, and \mathbf{H}_s is the $s \times s$ Hadamard matrix, defined recursively as

$$\mathbf{H}_2 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}, \quad \mathbf{H}_s = \begin{bmatrix} \mathbf{H}_{s/2} & \mathbf{H}_{s/2} \\ \mathbf{H}_{s/2} & -\mathbf{H}_{s/2} \end{bmatrix}. \quad (19)$$

The following theorem provides some guarantees about the FCT.

Theorem 3 ([9]) *Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ be a matrix of rank r ($r \ll n$). There is a distribution (given by the above FCT construction) over matrices $\mathbf{\Pi} \in \mathbb{R}^{h \times m}$ with $h = O(r \log r + r \log \frac{1}{\eta})$ such that, for all $\mathbf{x} \in \mathbb{R}^n$, the inequalities*

$$\|\mathbf{Ax}\|_1 \leq \|\mathbf{\Pi Ax}\|_1 \leq \tau \|\mathbf{Ax}\|_1 \quad (20)$$

hold with probability $1 - \eta$, where

$$\tau = O\left(\frac{r\sqrt{s}}{\eta} \log(hr)\right). \quad (21)$$

Considering η as a (small) constant and recalling that $s = h^6$, we have $h = O(r \log r)$ and finally $\kappa = O(r^4 \log^4 r)$. However, if we set $s \ll h^6$, the performance in practice does not seem to be negatively affected, even if this setting does not follow the above theorem [9, Section 6.1]. In our experiments, we use $s = h$.

3.1 Fast ℓ_1 regression

The FCT can be used as a building block for the construction of fast solvers for ℓ_1 regression problems. We describe this first before presenting the L1-NMF proposed algorithm.

Definition 4 ([26]) Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ be a matrix of rank r . A basis $\mathbf{B} \in \mathbb{R}^{n \times r}$ for the range of \mathbf{A} is (α, β) -conditioned if $\|\mathbf{B}\|_1 \leq \alpha$ and $(\forall \mathbf{x} \in \mathbb{R}^r) \|\mathbf{x}\|_\infty \leq \|\mathbf{B}\mathbf{x}\|_1$. We say that \mathbf{B} is well conditioned if α and β are low degree polynomials in r , independent of m and n .

Definition 5 ([26]) Given a well-conditioned basis \mathbf{B} for the range of $\mathbf{A} \in \mathbb{R}^{m \times n}$, we define the ℓ_1 leverage scores of \mathbf{A} as the m -dimensional vector λ , with elements $(\lambda)_i = \|(\mathbf{B})_{:i}\|_1$.

The leverage scores of \mathbf{A} can be found with the following procedure [9, 26]:

1. build an FCT matrix $\mathbf{\Pi} \in \mathbb{R}^{r \times m}$;
2. find a matrix \mathbf{R} such that $\mathbf{\Pi}\mathbf{A}\mathbf{R}$ is orthonormal;
3. using Definition 5, compute the leverage scores λ of $\mathbf{B} = \mathbf{A}\mathbf{R}^\dagger$ (\mathbf{R}^\dagger denotes the pseudoinverse of \mathbf{R});

The leverage scores are used in [9, 26] to speed up the algorithmic solution of $\mathbf{x}^* = \arg\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_1$. For this, the authors build a diagonal matrix \mathbf{E} , i.e., a compression matrix, by sampling its diagonal entries independently according to a set of probabilities p_i that are proportional to the ℓ_1 leverage scores of \mathbf{A} . Then, given $\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{E}(\mathbf{A}\hat{\mathbf{x}} - \mathbf{b})\|_1$ we have that $\|\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}\|_1 \leq (1 + \epsilon) \|\mathbf{A}\mathbf{x}^* - \mathbf{b}\|_1$ for some small ϵ . Key to the speedup is the size reduction of the regression problem, since only a few rows of \mathbf{A} are kept in $\mathbf{E}\mathbf{A}$ and considered to find $\hat{\mathbf{x}}$.

3.2 Fast ℓ_1 NMF

We now have all the elements that we need to build a fast algorithm for solving Problem (L1-NMF). Considering that we are dealing with a biconvex problem, it is a standard practice to find a solution to it by alternating two ℓ_1 least squares problems,

$$\hat{\mathbf{u}} = \arg \min_{\mathbf{u}} \|\mathbf{A} - \mathbf{u}\hat{\mathbf{v}}^T\|_1 \quad \text{s.t.} \quad \mathbf{u} \geq \mathbf{0}, \quad (22a)$$

$$\hat{\mathbf{v}} = \arg \min_{\mathbf{v}} \|\mathbf{A} - \hat{\mathbf{u}}\mathbf{v}^T\|_1 \quad \text{s.t.} \quad \mathbf{v} \geq \mathbf{0}. \quad (22b)$$

Each sub-problem can be sped up by respectively compressing the rows and the columns of \mathbf{A} . Similarly as in the regression case, row (resp. column) compression is achieved through the computation of the leverage scores of \mathbf{A} (resp. \mathbf{A}^T). Let \mathbf{C} and \mathbf{R} be the matrices that perform row and column compression, respectively. We can then iterate the *compressed* least squares problems

$$\hat{\mathbf{u}} = \arg \min_{\mathbf{u}} \|(\mathbf{A} - \mathbf{u}\hat{\mathbf{v}}^T)\mathbf{C}\|_1 \quad \text{s.t.} \quad \mathbf{u} \geq \mathbf{0}, \quad (23a)$$

$$\hat{\mathbf{v}} = \arg \min_{\mathbf{v}} \|\mathbf{R}(\mathbf{A} - \hat{\mathbf{u}}\mathbf{v}^T)\|_1 \quad \text{s.t.} \quad \mathbf{v} \geq \mathbf{0}. \quad (23b)$$

This would already give a very efficient algorithm for solving the problem of interest. In our experiments, we found that augmenting the procedure described in Page 7 with the above described compression techniques was faster and produced good results. We thus obtain the following *accelerated* procedure for solving Problem (L1-NMF):

1. Given a column compression matrix \mathbf{C} for \mathbf{A} , find an initialization for \mathbf{u} by solving the *compressed* problem

$$\hat{\mathbf{u}}, \hat{\mathbf{v}} = \underset{\mathbf{u}, \tilde{\mathbf{v}}}{\operatorname{argmin}} \|\mathbf{A}\mathbf{C} - \mathbf{u}\tilde{\mathbf{v}}^T\|_1 \quad \text{s.t.} \quad \mathbf{u} \geq \mathbf{0}, \tilde{\mathbf{v}} \geq \mathbf{0}, \quad (24)$$

with the iterative re-weighting scheme in [15]. The compressed vector $\hat{\mathbf{v}}$ has no further use in our algorithm.

2. Given the initialization $\hat{\mathbf{u}}$ and a row compression matrix $\mathbf{R}_{\hat{\mathbf{u}}}$ for $\mathbf{D}_{\hat{\mathbf{u}}}\mathbf{A}$ ($\mathbf{D}_{\mathbf{x}}$ is a diagonal matrix with the vector $\mathbf{1}_{[\mathbf{x}>0]}$ in its diagonal.), solve the *compressed* convex problem $\min_{\mathbf{v} \geq \mathbf{0}} \|\mathbf{R}_{\hat{\mathbf{u}}}\mathbf{D}_{\hat{\mathbf{u}}}(\mathbf{A} - \hat{\mathbf{u}}\mathbf{v}^T)\|_1$, using the ADMM technique in [30, App. A].
3. Given the new value of \mathbf{v} and a column compression matrix $\mathbf{C}_{\mathbf{v}}$ for $\mathbf{A}\mathbf{D}_{\mathbf{v}}$, solve the *compressed* convex problem $\min_{\mathbf{u} \geq \mathbf{0}} \|(\mathbf{A} - \mathbf{u}\mathbf{v}^T)\mathbf{D}_{\mathbf{v}}\mathbf{C}_{\mathbf{v}}\|_1$, using the same technique as before.

All compression matrices are computed with the same compression level r , used to build the FCT matrix. We also found in our experimental results that instead of using random sampling for building the row and column compression matrices, as in [9], good results were obtained by simply selecting the r largest leverage scores.

4. Experimental results

In the following, we refer to the proposed Random Sample Ensemble as RSE and to its compressed version, Accelerated Random Sample Ensemble, as ARSE. In our experiments, we compared the quantitative results of both methods, but only present qualitative results for ARSE. All the parameters for RSE and ARSE are exactly the same and in each figure/table we specify δ (Equation (4)); unless specified, $\kappa = 3$ (Definition 2). The only exception is the compression rate in ARSE, which will be set to $h = 32$ in all experiments, see Equation (17).

Evaluation. To measure performance, we use the standard precision and recall. In order to compute these measures, we compute an optimal assignment between the ground truth and the tested groups, using the Hungarian algorithm. Once the two sets of groups are appropriately matched, we can then compute the precision and the recall of the tested groups in a standard fashion.

If models are not allowed to share elements, it is not unusual in the literature to consider the outliers as an *additional* ground truth group to recover; in this case, the *misclassification error* is often reported.

In the general case where models overlap, we also use, as an additional measure, the generalized normalized mutual information (GNMI) [16], which extends the normalized mutual information (also called symmetric uncertainty) [33, p. 310] to the case of groups with overlaps.

2D lines and circles. We start our experimental evaluation with a few small synthetic datasets [31] where 2D points are arranged forming lines and circles. The results are shown in figures 3 and 4, where ARSE clearly detects the correct models in each case. In Table 1 we compare the performance of RSE and ARSE in all these datasets. As expected, when the dataset (and the preference matrix) are rather small, ARSE is only slightly faster than

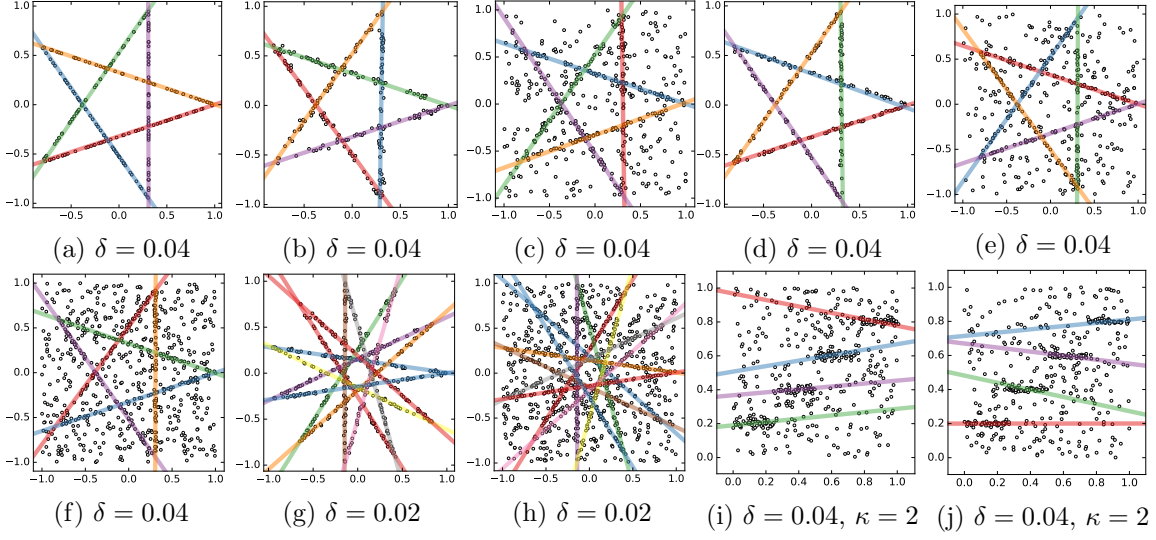


Figure 3: ARSE results for 2D lines detection. We show with different colors the final models estimated from the extracted biclusters (some colors may repeat themselves).

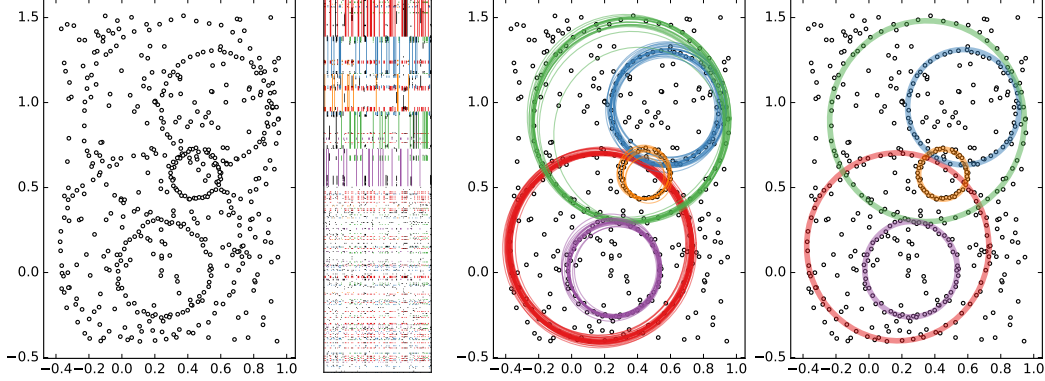


Figure 4: ARSE results for 2D circle detection on a synthetic dataset [31]. From left to right, original dataset, preference matrix, models retained in each bicluster, and models estimated from each bicluster. Each model/bicluster is denoted with a different color (some colors may repeat themselves). Notice that in the biclustered preference matrix, the models actually overlap. For this example, we set $\delta = 0.02$.

RSE; however, as the dataset gets larger (e.g., Figure 3(h)), ARSE becomes significantly faster than RSE.

Regarding the accuracy of both methods, RSE and ARSE are virtually equal for the examples in Figure 3, except in figures 3(i) and 3(j), where RSE performs slightly better. We also observe in figures 3 and 4 that the proposed approach can correctly recover overlapping models. This is an intrinsic limitation of previous state-of-the-art competitors such as J-linkage [31], T-linkage [20], RPA [21] and most multiple model estimation techniques, which are generally based on partitioning (clustering) the dataset.

Table 1: Comparative performance of RSE and ARSE for 2D lines and 2D circles detection. The datasets (figures 3 and 4) were created [31] by sampling points from different parametric models, adding a certain amount of noise to these points, and then further sampling outliers from a uniform distribution. Both methods, RSE and ARSE, perform well on these datasets. RSE results are slightly better, while ARSE is faster.

(a) The original ground truth does not consider model intersections (overlaps) nor “false” outliers that are actually located very near some model. This issue artificially affects the methods’ precision.

	Pref. matrix		Time (s)		GNMI		Precision		Recall	
	m	n	RSE	ARSE	RSE	ARSE	RSE	ARSE	RSE	ARSE
Figure 3(a)	250	1277	3.26	1.97	0.770	0.774	0.851	0.854	1.000	1.000
Figure 3(b)	250	1004	3.34	2.17	0.728	0.722	0.837	0.832	0.984	0.984
Figure 3(c)	500	692	3.32	1.68	0.678	0.634	0.723	0.698	1.000	0.984
Figure 3(d)	250	1133	3.06	1.93	0.794	0.785	0.867	0.861	1.000	1.000
Figure 3(e)	500	687	3.70	1.48	0.683	0.686	0.729	0.732	1.000	1.000
Figure 3(f)	750	557	3.53	1.53	0.639	0.636	0.655	0.651	1.000	1.000
Figure 3(g)	550	1080	13.17	4.16	0.698	0.697	0.759	0.761	0.987	0.984
Figure 3(h)	1100	668	14.13	3.60	0.651	0.644	0.657	0.654	0.993	0.989
Figure 3(i)	500	917	6.53	2.05	0.639	0.576	0.699	0.654	0.990	0.970
Figure 3(j)	500	1153	7.72	2.04	0.619	0.604	0.660	0.644	1.000	1.000
Figure 4	500	144	1.43	1.00	0.627	0.627	0.666	0.666	1.000	1.000
Mean			5.74	2.15	0.684	0.671	0.737	0.728	0.996	0.992
STD			4.27	0.93	0.059	0.068	0.081	0.086	0.006	0.010
Median			3.53	1.97	0.678	0.644	0.723	0.698	1.000	1.000

(b) We reestimate the ground truth by accounting for intersections and “false” outliers. As we can see, the precision is now as high as the recall.

	Pref. matrix		GNMI		Precision		Recall	
	m	n	RSE	ARSE	RSE	ARSE	RSE	ARSE
Figure 3(a)	250	1277	0.984	0.976	0.997	0.997	0.997	0.993
Figure 3(b)	250	1004	0.933	0.919	0.981	0.975	0.987	0.987
Figure 3(c)	500	692	0.968	0.852	0.986	0.939	0.997	0.968
Figure 3(d)	250	1133	0.985	0.928	0.993	0.976	1.000	0.990
Figure 3(e)	500	687	0.962	0.945	0.986	0.982	0.995	0.986
Figure 3(f)	750	557	0.961	0.905	0.990	0.968	0.990	0.972
Figure 3(g)	550	1080	0.974	0.956	0.992	0.99	0.994	0.986
Figure 3(h)	1100	668	0.950	0.945	0.980	0.977	0.989	0.988
Figure 3(i)	500	917	0.819	0.633	0.943	0.838	0.946	0.880
Figure 3(j)	500	1153	0.839	0.705	0.926	0.862	0.955	0.913
Figure 4	500	144	0.936	0.936	0.979	0.979	0.988	0.988
Mean			0.937	0.882	0.978	0.953	0.985	0.968
STD			0.056	0.111	0.022	0.053	0.018	0.037
Median			0.961	0.928	0.986	0.976	0.990	0.986

Table 2: Comparative performance of RSE and ARSE to detect vanishing points on the York Urban database [11]. The ground truth assumes a Manhattan world; hence, it only contains 2 or 3 vanishing points per image. This artificially affects the performance of the proposed method, since some images actually contain more than 3 vanishing points (see Figure 5).

	Time (s)		GNMI		Precision		Recall	
	RSE	ARSE	RSE	ARSE	RSE	ARSE	RSE	ARSE
Mean	3.66	1.52	0.735	0.695	0.853	0.833	0.868	0.852
STD	6.02	0.69	0.155	0.151	0.161	0.155	0.149	0.139
Median	2.09	1.42	0.764	0.706	0.939	0.916	0.938	0.915

Vanishing points. Under the assumption of perspective projection (e.g., with a pin-hole camera), sets of parallel lines in 3D space are projected into a 2D image as a set of concurrent lines. The point in the image plane at which these lines meet is called a *vanishing point*. Vanishing points provide important information to make inferences about the 3D structures of a scene from its 2D (projected) image. Since these projections are non-invertible, concurrence in the image plane does not necessarily imply parallelism in 3D. This said, the counterexamples for this implication are rare in real images, and the problem of finding parallel lines in 3D is reduced in practice to finding vanishing points in the image plane [e.g., 3, 4, 11, 18, 28]. We are here interested in finding vanishing points when no a priori information is known about the image or the camera with which it was taken.

The dataset elements in this case are line segments extracted with a suitable algorithm (e.g., [14]). As the distance between a vanishing point and a line segment, we consider the angular difference between the line segment and the line joining the vanishing point and the segment’s midpoint.

To evaluate the proposed framework for the task of detecting vanishing points, we use the York Urban database [11], which comes with ground truth assignments of image segments to three orthogonal directions in 3D. Of course, since the dataset images represent urban scenes, performance can be readily boosted by adding problem-specific constraints to the proposed algorithm as an extra postprocessing step; common simplifications include knowledge about the camera calibration and the Manhattan world assumption [e.g., 18].

The results of the proposed method are presented in Table 2. There we can observe that RSE and ARSE perform well in terms of precision and recall. ARSE is in average twice as fast as RSE. In Figure 5 we can observe a few characteristic examples of ARSE’s results. Notice that in the two rightmost examples, the method correctly finds more than three vanishing points; these additional points are deemed incorrect in the results of Table 2, as the ground truth only contemplates three points.

Fundamental matrices and homographies. For these examples, where we are trying to estimate geometric transforms between two images, the base elements are keypoint matches. Any modern method to find those matches would work relatively well for our purposes; we use BRISK [17]. We show in Figure 6 two examples of fundamental matrix estimation on a stereo pair of images. The only motion in the scene corresponds to a change in camera perspective, and it can be described with a single fundamental matrix. Both

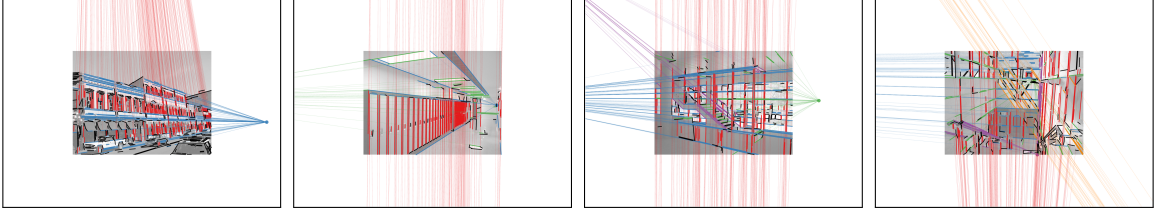


Figure 5: Examples of vanishing points detection on the York Urban database [11]. ARSE can detect any number of vanishing points (from left to right, 2, 3, 4, and 5 detections, respectively) , even when their directions are not orthogonal in 3D. For this experiment, we set $\delta = 10^{-2}\pi$.

methods achieve extremely similar results and in both cases (left and right pairs) they find a single model. The example on the left contains 2518 matches. ARSE detects 1377 matches in 7.99 seconds, while RSE detects 1363 matches in 98.32 seconds. ARSE’s precision and recall, considering the result of RSE as ground truth, are 0.959 and 0.968, respectively. The example on the right contains 8999 matches. ARSE detects 5533 matches in 30.40 seconds, whereas RSE detects 5537 matches in 1033.45 seconds. In this case, the precision is 0.959 and recall is 0.968. However, there is a huge difference in speed between both methods, which is accentuated with the size of the dataset. On the left example, ARSE is about 12 times faster, whereas on the right example it is about 34 times faster.

It is important to emphasize that other random sampling techniques can be used beyond the baseline in Algorithm 1. To show this, we have included experiments using the state-of-the-art MultiGS algorithm [7]. It is worth pointing out that a recent technique [5] has been able to reduce the size of the minimal sample set for fundamental matrices to $b = 2$; its use would significantly reduce the computational complexity of the sampling step, imposing an upper bound of $O(m^2)$ to the total number of possible combinations.

We estimate multiple fundamental matrices (moving camera and moving objects) and multiple homographies (moving planar objects) on the images in the AdelaideRMF dataset [34]. Although we use this dataset because it is standard in the literature [e.g., 20, 21, 23, 34], we would like to point out that the ground truth of this dataset contains a non-negligible quantity of severe errors. In Figure 7 we show two examples where these errors are easily identified upon visual inspection. The dataset’s ground truth would need a thorough revision to be truly useful as a scientific benchmark. We include these results nonetheless for the sake of completeness.

The results on the full dataset are presented in Figure 8. We include comparisons with T-linkage [20], RCMSA [23], and RPA [21]. Among these three methods, RPA is the one that achieves the best performances; however, to yield these performances RPA uses as an input the *ground truth* number of models to estimate. The other methods, as the proposed method, automatically estimate this number.

In Figure 8(a), RSE and ARSE perform on average similarly as T-linkage. However, notice that the median performance of ARSE is similar to the one of RPA, with the added benefit of not having to tune a critical and fundamental parameter. In Figure 8(b), both RSE and ARSE clearly outperform all other methods. The complete numerical results are included in tables 3 and 4.



Figure 6: Fundamental matrix estimation on two stereo pairs of the Middlebury Stereo dataset [25]. The stereo pairs are calibrated such that all epipolar lines are horizontal. Since the scene is rigid, a single fundamental matrix is enough to describe the stereo geometry. In both cases, a single model (fundamental matrix) was found by both ARSE and RSE. From top to bottom: original pair, matches obtained from BRISK keypoints and descriptors [17] (2518 on the left, 8999 on the right), ARSE inliers (1363/2518 \approx 54% on the left, 5537/8999 \approx 61% on the right), and ARSE outliers (the RSE inlier sets are very similar). As expected, the lines connecting inlier matches are horizontal.

3D planes estimation. In recent years, people have started “scanning” objects with range imaging hardware (e.g., Kinect or RealSense cameras) to obtain 3D point clouds. The geometric nature of man-made objects (composed of planes, cylinders, spheres, etc.) is very well adapted to the analysis of their corresponding point clouds with MPME.

As an example, we show the capabilities of our method in this setting with the detection of 3D planes on the Pozzovegianni dataset.² In this case, the 3D points are obtained from different images of a building (Figure 9(a)) with a sparse multi-view 3D reconstruction algorithm. Our method recovers the 3D planes in the scene and properly reconstructs the building structure. We evaluate our results on four different versions of the dataset, using 10%, 20%, 50%, and 100% the points. In Figure 10(a), we show the estimated planes on the first (10%) and last (100%) versions. Both results are very similar except for the bell tower, which is correctly recovered in the latter case but not in the former (mainly because there are not enough points for a proper estimation).

2. <http://www.diegm.uniud.it/fusiello/demo/samantha/>

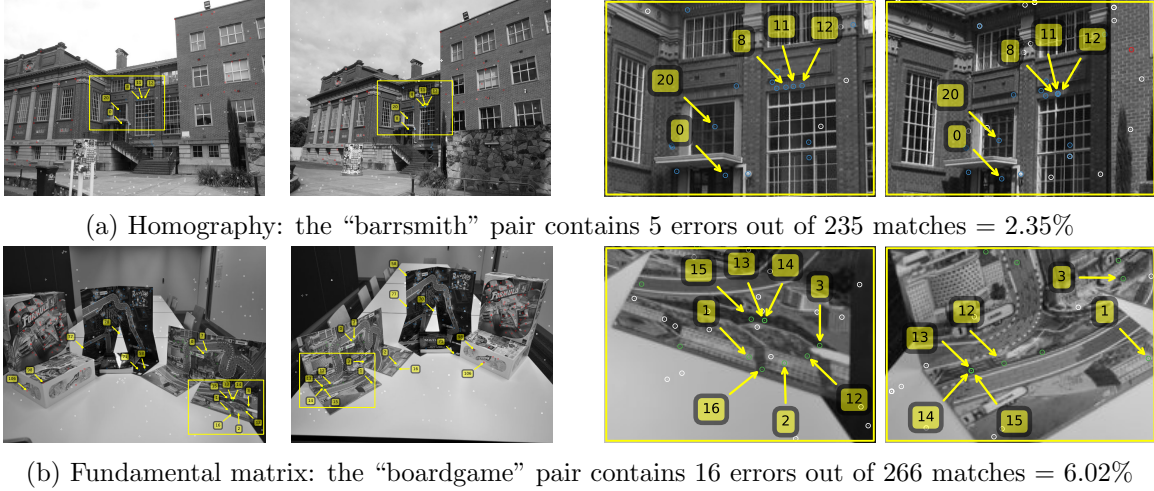


Figure 7: The ground truth of the standard AdelaideRMF dataset [34] has non-negligible errors (detectable by simple ocular inspection). We provide two examples of such errors, one for each type of geometric model. On each example, we show the original pair on the left, and provide a zoom-in to a specific region on the right. Inliers are indicated in color and outliers in white; the wrong matches are identified with matching numbers.

The experimental setup used in the above paragraph provides a good opportunity to measure the speed difference between RSE and ARSE, see Figure 11(a). ARSE is faster than RSE for small datasets, as in all the previous examples; however, the speed difference is not as striking (2.4 times faster when using 10% of the points). When working with increasingly bigger datasets, this difference becomes more and more pronounced (60 times faster when using 100% of the points), see the left plot in Figure 11(a). The other appealing quality of ARSE is that its speed scales very gracefully with the preference matrix size, see the right plot in Figure 11(a).

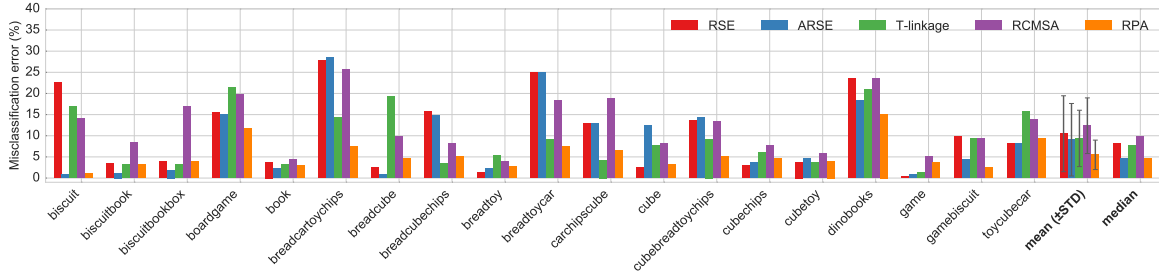
We ran the same experiment on the Piazza Bra dataset,³ see Figure 12. We only ran RSE on the dataset versions that include 10% and 20% of the points, because of RAM memory issues. Notice that ARSE with a 82831×24841 preference matrix is faster than RSE with a 16567×2040 preference matrix (the former matrix being about 61 times bigger than the latter).

5. Conclusions

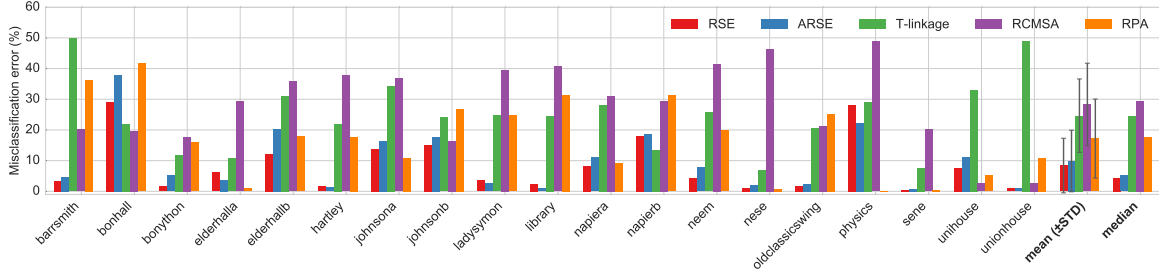
In this work we introduced a complete and comprehensive algorithmic pipeline for multiple parametric model estimation. The proposed approach takes the information produced by a random sampling algorithm (e.g., RANSAC) and analyzes it from a machine learning/optimization perspective, using a *parameterless* biclustering algorithm based on L1 nonnegative matrix factorization (L1-NMF).

This new formulation conceptually changes the way that the data produced by the popular RANSAC, or related model-candidate generation techniques, is analyzed. It exploits

3. See Footnote 2



(a) Fundamental matrices: $\delta = 3$



(b) Homographies: $\delta = 14.5$

Figure 8: Results on the AdelaideRMF dataset [34] for the estimation of fundamental matrices and homographies. Note that RPA does not automatically estimate the number of models (the ground truth number of models was used in all examples). In each subfigure, we indicate the value chosen for δ (Equation (4)).

consistencies that naturally arise during the RANSAC execution, while explicitly avoiding spurious inconsistencies. Additionally and contrarily to the main trends in the literature, the proposed modeling does not impose non-intersecting parametric models.

We also presented an accelerated version of the biclustering process, introducing an accelerated algorithm to solve L1-NMF problems. This allows to solve medium-sized problems faster while also extending the usability of the algorithm to much larger datasets. We emphasize that this contribution exceeds the context of this work, as this accelerated algorithm has potential applications in any other context where an L1-NMF is needed.

As a future line of research, we are currently investigating whether using a hard thresholding scheme is actually necessary. Instead of working with binary data, we could work with a real-valued object-model distance matrix, eliminating a critical parameter that has been haunting the RANSAC framework for years.

References

- [1] N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *STOC*, 2006.
- [2] N. Ailon and B. Chazelle. The fast Johnson-Lindenstrauss transform and approximate nearest neighbors. *SIAM J. Comput.*, 39(1):302–322, 2009.

Table 3: Results on the AdelaideRMF dataset [34] for the estimation of multiple fundamental matrices, see Figure 8(a).

	Time (s)		Misclassification error				
	RSE	ARSE	RSE	ARSE	T-linkage [20]	RCMSA [23]	RPA [21]
biscuit	2.18	1.23	22.60	0.90	16.93	14.00	1.15
biscuitbook	3.09	1.31	3.50	1.20	3.23	8.41	3.23
biscuitbookbox	2.34	0.99	3.90	1.90	3.10	16.92	3.88
boardgame	1.76	0.56	15.40	15.00	21.43	19.80	11.65
book	4.09	1.09	3.80	2.20	3.24	4.32	2.88
breadcartoychips	0.39	0.29	27.70	28.60	14.29	25.69	7.50
breadcube	4.13	1.42	2.60	0.90	19.31	9.87	4.58
breadcubechips	1.20	0.89	15.70	14.80	3.48	8.12	5.07
breadtoy	5.75	1.55	1.40	2.20	5.40	3.96	2.76
breadtoycar	0.16	0.17	25.00	25.00	9.15	18.29	7.52
carchipscube	1.10	0.46	12.80	12.80	4.27	18.90	6.50
cube	1.12	0.63	2.40	12.50	7.80	8.14	3.28
cubebreadtoy chips	1.56	1.02	13.70	14.30	9.24	13.27	4.99
cube chips	1.11	0.57	2.90	3.60	6.14	7.70	4.57
cubetoy	2.11	0.87	3.80	4.60	3.77	5.86	4.04
dinobooks	2.75	0.96	23.60	18.30	20.94	23.50	15.14
game	0.32	0.26	0.40	0.90	1.30	5.07	3.62
gamebiscuit	2.09	0.87	9.90	4.30	9.26	9.37	2.57
toycubecar	1.03	0.67	8.10	8.10	15.66	13.81	9.43
Mean	2.01	0.83	10.48	9.06	9.37	12.37	5.49
STD	1.45	0.39	8.97	8.57	6.66	6.58	3.48
Median	1.76	0.87	8.10	4.60	7.80	9.87	4.57

- [3] A. Almansa, A. Desolneux, and S. Vamech. Vanishing point detection without any a priori information. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(4):502–507, 2003.
- [4] F. Andaló, G. Taubin, and S. Goldenstein. Vanishing point detection by segment clustering on the projective space. In *ECCV Workshops*, 2012.
- [5] G. Ben-Artzi, T. Halperin, M. Werman, and S. Peleg. Two points fundamental matrix. Technical report, 2016. URL <http://arxiv.org/abs/1604.04848>.
- [6] N. Burrus, T. M. Bernard, and J. M. Jolion. Image segmentation by a contrario simulation. *Pattern Recognit.*, 42(7):1520–1532, 2009.
- [7] T. J. Chin, J. Yu, and D. Suter. Accelerated hypothesis generation for multistructure data via preference analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(4):625–638, 2012.
- [8] S. Choi, T. Kim, and W. Yu. Performance evaluation of RANSAC family. In *BMVC*, 2009.
- [9] K. L. Clarkson, P. Drineas, M. Magdon-Ismail, M. W. Mahoney, X. Meng, and D. P. Woodruff. The fast Cauchy transform and faster robust linear regression. *SIAM J. Comput.*, 45(3):763–810, 2016.

Table 4: Results on the AdelaideRMF dataset [34] for the estimation of multiple homographies, see Figure 8(b).

	Time (s)		Misclassification error				
	RSE	ARSE	RSE	ARSE	T-linkage [20]	RCMSA [23]	RPA [21]
barrsmith	1.65	2.18	3.40	4.70	49.79	20.14	36.31
bonhall	90.37	13.50	29.20	38.00	21.84	19.69	41.67
bonython	1.93	1.60	1.60	5.20	11.92	17.79	15.89
elderhalla	3.37	2.37	6.10	3.70	10.75	29.28	0.93
elderhallb	6.08	4.86	12.20	20.40	31.02	35.78	17.82
hartley	6.14	3.53	1.60	1.30	21.90	37.78	17.78
johnsona	22.55	7.86	13.90	16.40	34.28	36.73	10.76
johnsonb	81.41	0.93	15.10	17.60	24.04	16.46	26.76
ladysymon	11.30	5.70	3.50	2.60	24.67	39.50	24.67
library	4.03	2.66	2.40	0.90	24.53	40.72	31.29
napiera	4.83	2.98	8.20	11.00	28.08	31.16	9.25
napierb	11.34	4.88	18.10	18.60	13.50	29.40	31.22
neem	9.51	2.73	4.30	7.80	25.65	41.45	19.86
nese	13.05	6.21	1.20	2.10	7.05	46.34	0.83
oldclassicswing	15.60	6.07	1.70	2.50	20.66	21.30	25.25
physics	2.93	3.38	28.20	22.30	29.13	48.87	0.00
sene	8.22	4.03	0.40	0.80	7.63	20.20	0.42
unihouse	229.62	17.10	7.60	11.20	33.13	2.56	5.21
unionhouse	1.03	1.03	0.90	0.90	48.99	2.64	10.87
Mean	27.63	4.93	8.40	9.89	24.66	28.30	17.20
STD	55.02	4.14	8.90	10.04	11.96	13.45	12.87
Median	8.22	3.53	4.30	5.20	24.53	29.40	17.78

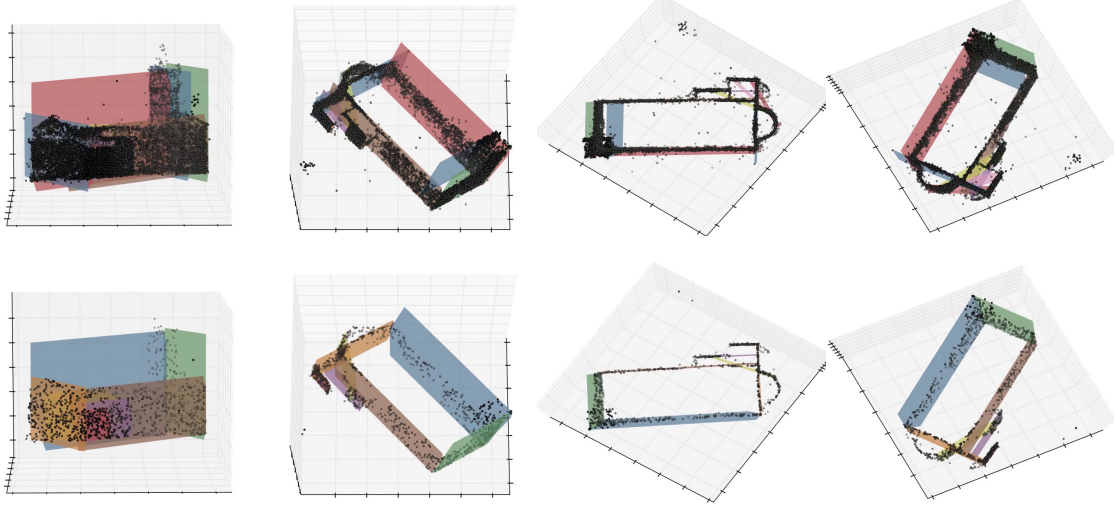
- [10] T. M. Cover. Enumerative source encoding. *IEEE Trans. Inf. Theory*, 19(1):73–77, 1973.
- [11] P. Denis, J. H. Elder, and F. Estrada. Efficient edge-based methods for estimating manhattan frames in urban imagery. In *ECCV*, 2008.
- [12] A. Desolneux, L. Moisan, and J. M. Morel. *From Gestalt Theory to Image Analysis*, volume 34. Springer-Verlag, 2008.
- [13] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.
- [14] R. Grompone von Gioi, J. Jakubowicz, J. M. Morel, and G. Randall. LSD: A fast line segment detector with a false detection control. *IEEE Trans Pattern Anal Mach Intell*, 32(4):722–732, 2010.
- [15] D. Kong, C. Ding, and H. Huang. Robust nonnegative matrix factorization using l21-norm. In *CIKM*, 2011.
- [16] A. Lancichinetti, S. Fortunato, and J. Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New J. Phys.*, 11(3):33015, 2009.

- [17] S. Leutenegger, M. Chli, and R. Y. Siegwart. BRISK: Binary robust invariant scalable keypoints. In *ICCV*, 2011.
- [18] J. Lezama, R. von Gioi, G. Randall, and J.-M. Morel. Finding vanishing points via point alignments in image primal and dual domains. In *CVPR*, 2014.
- [19] T. Li, C. Ding, and M. I. Jordan. Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization. In *ICDM*, 2007.
- [20] L. Magri and A. Fusiello. T-linkage: A continuous relaxation of J-linkage for multi-model fitting. In *CVPR*, 2014.
- [21] L. Magri and A. Fusiello. Robust multiple model fitting with preference analysis and low-rank approximation. In *BMVC*, 2015.
- [22] E. E. Papalexakis, N. D. Sidiropoulos, and R. Bro. From K-means to higher-way co-clustering: multilinear decomposition with sparse latent factors. *IEEE Trans. Signal Process.*, 61(2):493–506, 2013.
- [23] T. T. Pham, T. J. Chin, J. Yu, and D. Suter. The random cluster model for robust geometric fitting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(8):1658–1671, 2014.
- [24] J. Rabin, J. Delon, Y. Gousseau, and L. Moisan. MAC-RANSAC: A robust algorithm for the recognition of multiple objects. In *3DPTV*, 2010.
- [25] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *GCPR*, 2014.
- [26] C. Sohler and D. Woodruff. Subspace embeddings for the L1-norm with applications. In *STOC*, 2011.
- [27] R. Subbarao and P. Meer. Nonlinear mean shift for clustering over analytic manifolds. In *CVPR*, 2006.
- [28] J. P. Tardif. Non-iterative approach for fast and accurate vanishing point detection. In *ICCV*, 2009.
- [29] M. Tepper, A. Newson, P. Sprechmann, and G. Sapiro. Multi-temporal foreground detection in videos. In *ICIP*, 2015.
- [30] M. Tepper and G. Sapiro. A bi-clustering framework for consensus problems. *SIAM J. Imaging Sci.*, 7(4):2488–2525, 2014.
- [31] R. Toldo and A. Fusiello. Robust multiple structures estimation with J-linkage. In *ECCV*, 2008.
- [32] D. M. Witten, R. Tibshirani, and T. Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Bio-statistics*, 10(3):515–534, 2009.

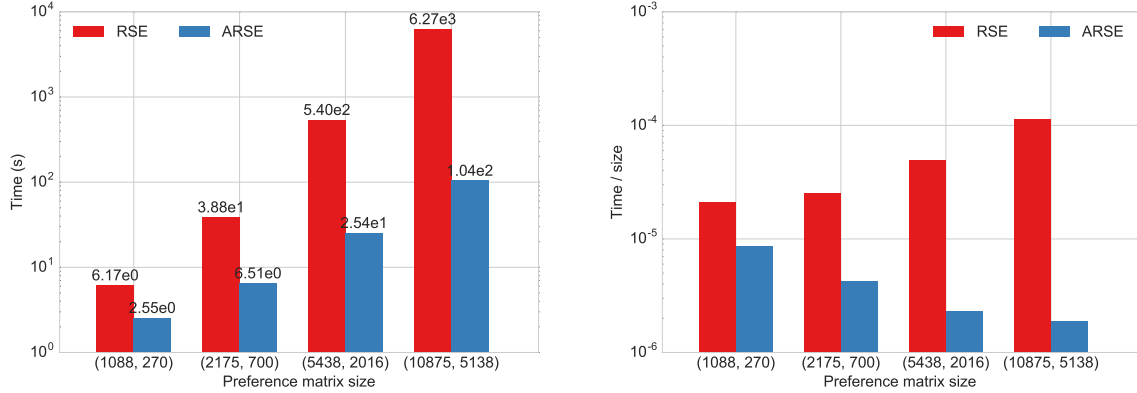
- [33] I. H. Witten, E. Frank, and M. a. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., 2011.
- [34] H. S. Wong, T. J. Chin, J. Yu, and D. Suter. Dynamic and hierarchical multi-structure geometric model fitting. In *ICCV*, 2011.
- [35] L. Xu, E. Oja, and P. Kultanen. A new curve detection method: Randomized Hough transform (RHT). *Pattern Recognit. Lett.*, 11(5):331–338, 1990.
- [36] R. Zhao and V. Y. F. Tan. Online Nonnegative Matrix Factorization with Outliers. Technical report, 2016. URL <http://arxiv.org/abs/1604.02634>.
- [37] M. Zuliani, C. S. Kenney, and B. S. Manjunath. The multiRANSAC algorithm and its application to detect planar homographies. In *ICIP*, 2005.



(a) A few samples of the original images used to build the 3D point cloud.



(a) The planes estimated by ARSE seen from different angles. We show on the top row the results using the original dataset (10875 points) and the dataset with 10x subsampling on the bottom row (1088 points).



(a) On the left, comparison between the running times of RSE and ARSE as the dataset size (and hence the preference matrix size) increase. As this progression continues, RSE becomes much faster than ARSE (notice the logarithmic scale). On the right, we plot the run-time divided by the preference matrix size, giving a common ground to compare the performance for different preference matrix sizes. An algorithm linear in the preference matrix size would exhibit a flat behavior in this plot; we can easily observe that while RSE is supralinear, ARSE is clearly sublinear.

Figure 11: 3D planes estimation. We run the whole algorithmic pipeline on four different versions of the Pozzoveggiani dataset, created by subsampling the original dataset 10, 5, 2, 1 times, yielding 1088, 2175, 5438, and 10875 3D points, respectively. For this experiment, we set $\delta = 10^{-1}$ (Equation (4)).

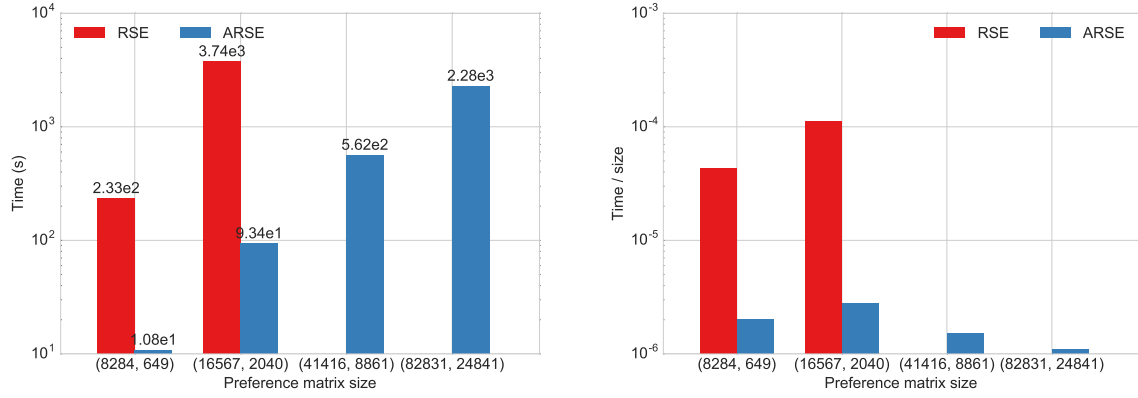


Figure 12: 3D planes estimation. Same experiment as in Figure 11 with the Piazza Bra dataset, created by subsampling the original dataset 10, 5, 2, 1 times, yielding 8284, 16567, 41414, and 82831 3D points, respectively. For this experiment, we set $\delta = 10^{-1}$ (Equation (4)).